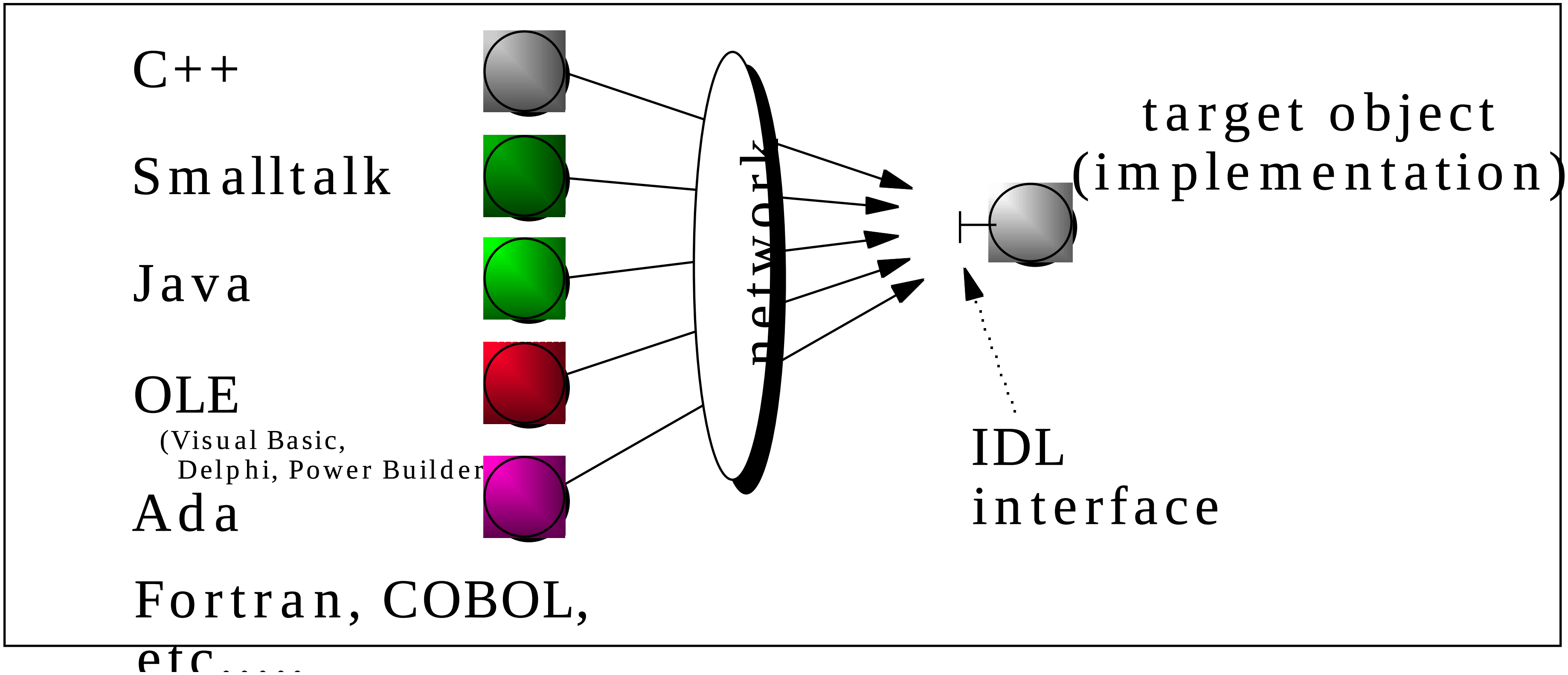
RTI CORBA Cap
Design Overview and Status

Dr. Glenn H. Tarbox
Object Sciences Corp

tarbox@objsci.com
www.objsci.com

CORBA Basics

- CORBA: Common Object Request Broker Architecture
  - A technology for integrating distributed systems across networks, operating systems, and languages
- IDL: Interface Definition Language
  - A language neutral representation of interfaces



C++

Smalltalk

Java

OLE
(Visual Basic,
Delphi, Power Builder

Ada

Fortran, COBOL,
etc.....

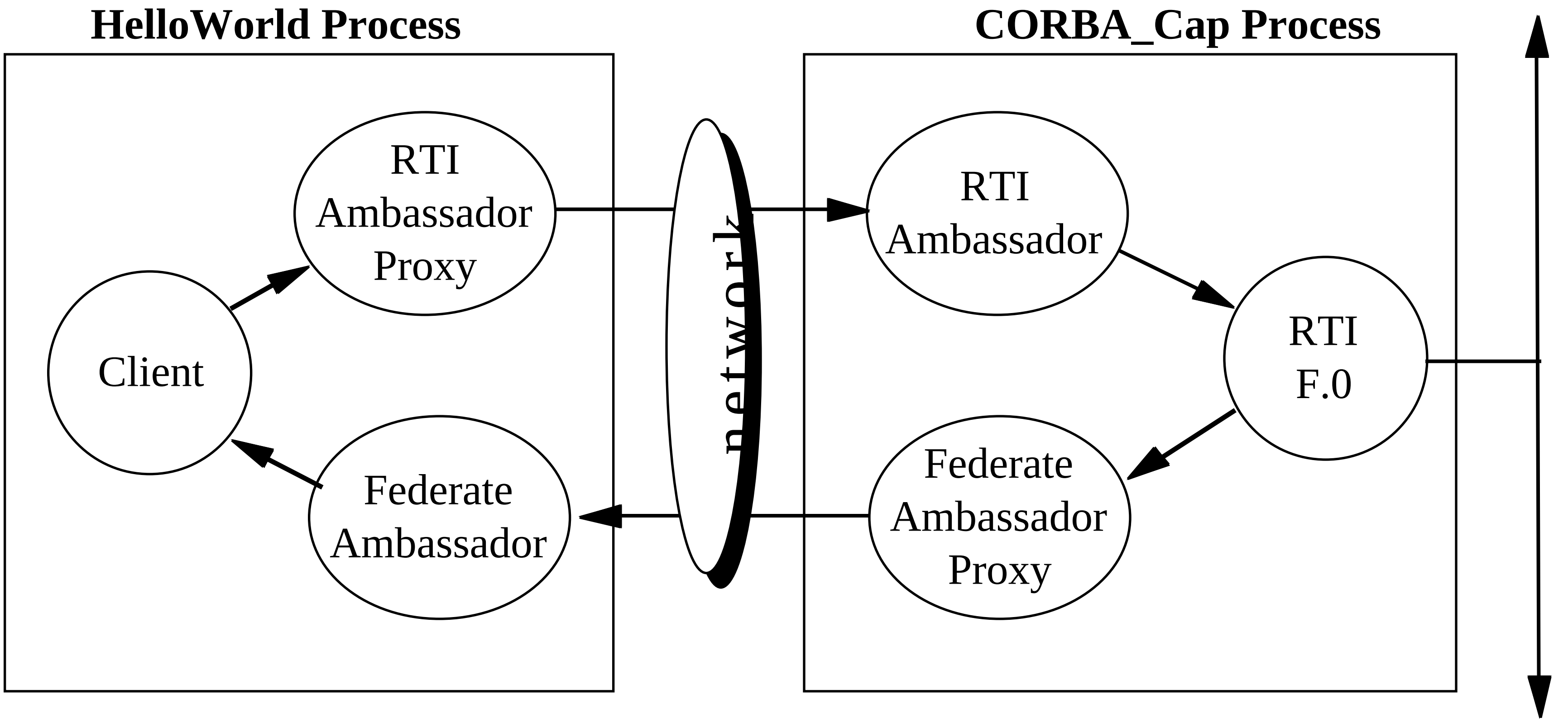
network

target object
(implementation)

IDL
interface

CORBA Cap Tasks

- ❑ Define IDL interface for RTI 1.0 Interface Specification
- ❑ Implement interface in C++ by *Wrapping* F.0 (Beta R8)
- ❑ Modify HelloWorld example to use IDL defined interface



RTI 1.0 IDL Interface: Operations

- C++ and IDL RTI interfaces map closely
  - IDL and C++ have similar syntax in general

```
FederateHandle                                                         // returned C3
joinFederationExecution (
  const FederateName                yourName,                          // supplied C4
  const FederationExecutionName   executionName,                      // supplied C4
      FederateAmbassadorPtr         federateAmbassadorReference)     // supplied C1
throw (
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  CouldNotOpenFED,
  ErrorReadingFED,
  ConcurrentAccessAttempted,
  RTIinternalError);
```

C++

```
FederateHandle
joinFederationExecution (
  in FederateName                 yourName,
  in FederationExecutionName   executionName,
  in FederateAmbassador          federateAmbassador)
raises (
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  CouldNotOpenFED,
  ErrorReadingFED,
  ConcurrentAccessAttempted,
  RTIinternalError, Deadlock);
```

IDL

RTI 1.0 Interface: Complex Types

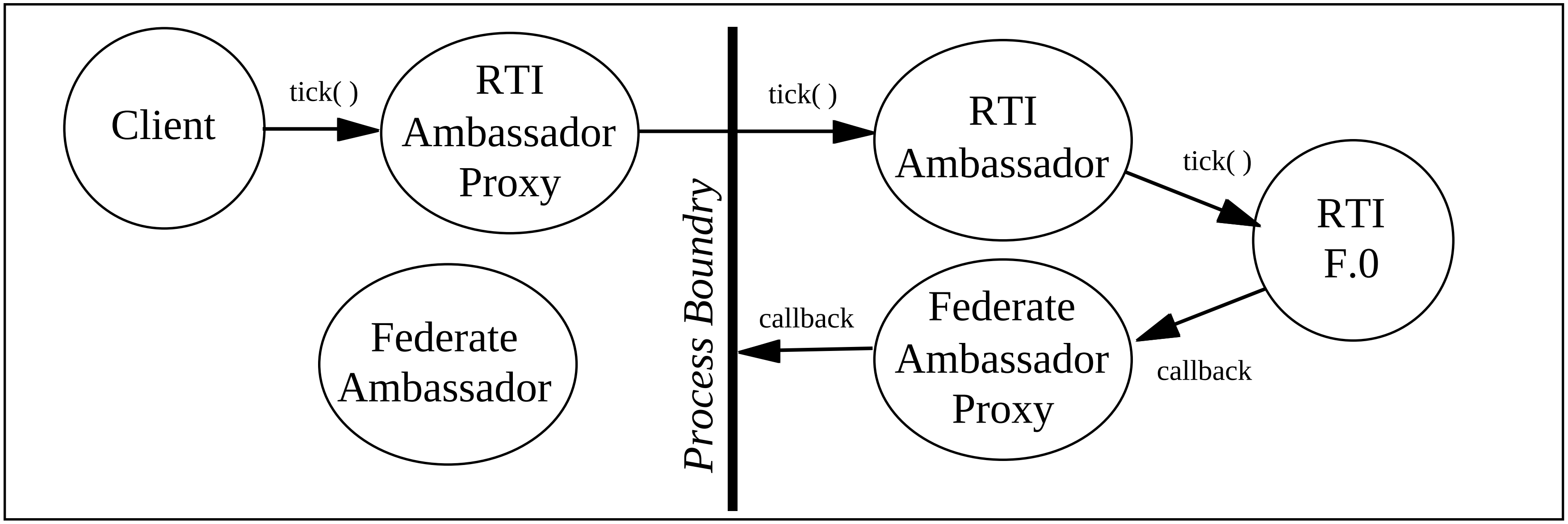
- Basic (atomic) types map simply
  - longs, shorts, doubles, strings, exceptions
- Complex types need more work….
  - sequences were simple to define in IDL but required translation for the implementation

IDL

```
typedef sequence<octet> AttributeValue;

struct AttributeHandleValuePair {
        AttributeHandle handle;
        AttributeValue value;
};

typedef sequence<AttributeHandleValuePair> AttributeHandleValuePairSet;

typedef sequence<AttributeHandle> AttributeHandleSet;
```

Tick and Deadlock

- F.0 assumes a single threaded client
  - tick( ) is used to give the RTI cycles
  - all callbacks to the federate occur “within” tick( )
- Single threaded “client” can’t service invocations when blocked on a synchronous call
  - federateAmbassador operations can’t be serviced during synchronous operations on RTIambassador



Asychronous Tick or
Multi-Threaded Server

- ❑ Solution #1: use asynchronous tick( )
  - – non-blocking tick( ) returns immediately
  - – when no callback, requires null message or timeout from RTI
  - – Lots of network traffic for no purpose
- ❑ Solution #2: Tick in separate thread
  - – thread ticks RTI at programmable rate
  - – client uses processEvents( ) to give thread to callback operations on federateAmbassador
- ❑ Threading requires synchronization of invocations on RTIambassador and thread in tick( )
  - – RTI is thread-safe but not re-entrant.
- ❑ Slight possibility of callback in progress during invocation from client
  - – immediately detected in server and Deadlock exception thrown

Single Threaded Client

- ❑ Interface Extensions
  - Operations

    beginTicking(in long sleepTime)

    endTicking( )
  - Exceptions

    Deadlock

    TickingPreviouslyStarted

    TickingNotEnabled
- ❑ Client catches *Deadlock* exception for each invocation
  - typically calls processEvents( ) to process callback and retries

Multi-Threaded Client

- ❑ Better approach is to implemement multi-threaded client
  - – set of threads to implement model behavior
  - – single thread to process invocations on <u>federateAmbassador</u>
- ❑ Deadlock exception not thrown
  - – as F.0 RTI is not re-entrant, synchronization of calls to RTIambassador and tick( ) still required.

CORBA Cap Status

- ❑ First version of Cap delivered to Virtual Technology Corp. for testing and packaging
- ❑ Interface for multi-threaded client in progress
  - – run time selection to disable Deadlock exceptions